Project 1

2024-03-19

Getting Started

Intro and concern

Acquiring thorough profit insights is critical in the current competitive online shopping platform scenario. These insights are crucial for developing successful sales tactics and offers, which in turn lead to higher sales and more profits.

We built a set of functions using linear regression techniques on top of Amazon sales data. With the use of these features, we can identify weekly profit trends depending on consumer actions. Our ability to accurately and precisely improve sales performance can be enhanced by utilizing this analytical method.

Question: Can we develop a profit function using the provided sales data that offers insights into profitability, considering various aspects such as purchase day, repeat purchases, and product categories?

Importing the Libraries

```
library(ggplot2)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.3.3
```

```
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.3.3
```

```
library(nnet)
library(modelr)
```

```
## Warning: package 'modelr' was built under R version 4.3.3
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3
```

```
## Warning: package 'readr' was built under R version 4.3.3
```

```
## Warning: package 'purrr' was built under R version 4.3.3
```

```
## Warning: package 'forcats' was built under R version 4.3.3
```

```
## Warning: package 'lubridate' was built under R version 4.3.3
```

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.3.3
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.3
```

```
library(knitr)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

Importing Data

Utilizing the 'readxl' library, we efficiently import Excel data, transforming it into a dataset labeled 'sales.data'. This dataset serves as a comprehensive repository of our sales information, ready for analysis and exploration.

Revised column names for improved accessibility.

```
sales.data <- read_excel("Amazon_2_Raw.xlsx")
colnames(sales.data) <- c("order_id","order_date","ship_date","email","geography","category","product_na
```

Overview of Data

Let's take a moment to examine the dataset and identify the most frequently accessed information, as well as areas with ample room for further analysis and exploration.

Summary of Sales Data

There are a total of 10 fields within our dataset: Order ID, Order Date, Ship Date, Email ID, Geography, Category, Product Name, Sales, Quantity, and Profit. These fields offer rich insights into customer behavior and transactional patterns. However, a minor issue arises with some missing data in the Order Date and Ship Date fields, potentially impacting the accuracy of our analysis. To mitigate this, we'll focus on alternative feilds, more reliable information to delve deeper into customer behavior and its correlation with profitability.

```
kable(summary(sales.data))
```

order_id	order_date	ship_date	email	geography	category	product_name	sales	quantity	profit
Length:3203	Min. :2011-01-07 00:00:00.00	Min. :2011-01-09 00:00:00.0	Length:3203	Length:3203	Length:3203	Length:3203	Min. : 0.99	Min. : 1.000	Min. :-3399.980
Class :character	1st Qu.:2012-05-22 00:00:00.00	1st Qu.:2012-05-26 00:00:00.0	Class :character	Class :character	Class :character	Class :character	1st Qu.: 19.44	1st Qu.: 2.000	1st Qu.: 3.852
Mode :character	Median :2013-07-22 00:00:00.00	Median :2013-07-25 00:00:00.0	Mode :character	Mode :character	Mode :character	Mode :character	Median : 60.84	Median : 3.000	Median : 11.166
NA	Mean :2013-05-10 03:06:07.52	Mean :2013-05-14 01:25:25.2	NA	NA	NA	NA	Mean : 226.49	Mean : 3.829	Mean : 33.849
NA	3rd Qu.:2014-05-23 00:00:00.00	3rd Qu.:2014-05-27 00:00:00.0	NA	NA	NA	NA	3rd Qu.: 215.81	3rd Qu.: 5.000	3rd Qu.: 33.000

order_id	order_date	ship_date	email	geography	category	product_name	sales	quantity	profit
NA	Max. :2014-12-31 00:00:00.00	Max. :2015-01-06 00:00:00.0	NA	NA	NA	NA	Max. :13999.96	Max. :14.000	Max. : 6719.981

```
kable(head(sales.data))
```

order_id	order_date	ship_date	email	geography	category	product_name	sales	quantity	profit
CA-2013-138688	2013-06-13	2013-06-17	DarrinVanHuff@gmail.com	United States,Los Angeles,California	Labels	Self-Adhesive Address Labels for Typewriters by Universal	14.620	2	6.8714
CA-2011-115812	2011-06-09	2011-06-14	BrosinaHoffman@gmail.com	United States,Los Angeles,California	Furnishings	Eldon Expressions Wood and Plastic Desk Accessories, Cherry Wood	48.860	7	14.1694
CA-2011-115812	2011-06-09	2011-06-14	BrosinaHoffman@gmail.com	United States,Los Angeles,California	Art	Newell 322	7.280	4	1.9656
CA-2011-115812	2011-06-09	2011-06-14	BrosinaHoffman@gmail.com	United States,Los Angeles,California	Phones	Mitel 5320 IP Phone VoIP phone	907.152	4	90.7152
CA-2011-115812	2011-06-09	2011-06-14	BrosinaHoffman@gmail.com	United States,Los Angeles,California	Binders	DXL Angle-View Binders with Locking Rings by Samsill	18.504	3	5.7825
CA-2011-115812	2011-06-09	2011-06-14	BrosinaHoffman@gmail.com	United States,Los Angeles,California	Appliances	Belkin F5C206VTEL 6 Outlet Surge	114.900	5	34.4700

Data Preprocessing and Feautre Engineering

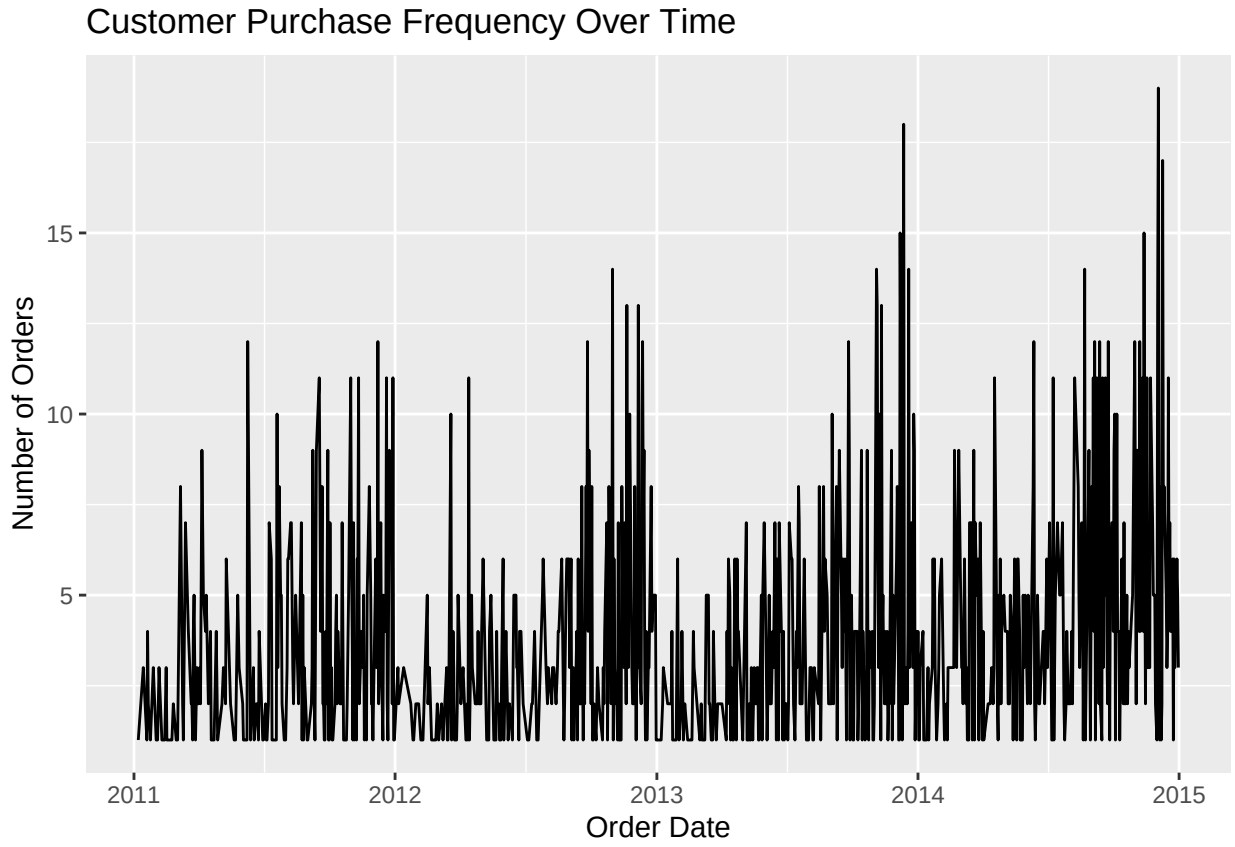
```
sales.data$order_date <- as.Date(sales.data$order_date)
sales.data$ship_date <- as.Date(sales.data$ship_date)
sales.data$year <- format(sales.data$order_date, "%Y")
sales.data$month <- format(sales.data$order_date, "%m")
sales.data$day_of_week <- weekdays(sales.data$order_date)
sales.data$shipping_duration <- as.numeric(difftime(sales.data$ship_date, sales.data$order_date, units =
sales.data$year <- as.factor(sales.data$year)
sales.data$month <- as.factor(sales.data$month)
sales.data$day_of_week <- as.factor(sales.data$day_of_week)
sales.data$category <- as.factor(sales.data$category)
sales.data$sales_normalized <- scale(sales.data$sales)
sales.data$quantity_normalized <- scale(sales.data$quantity)
sales.data$shipping_duration_normalized <- scale(sales.data$shipping_duration)
sales.data$order_day <- weekdays(as.Date(sales.data$order_date))
sales.data$repeat_purchases <- ave(sales.data$order_id, sales.data$email, FUN = length)
sales.data$repeat_purchases <- as.numeric(as.character(sales.data$repeat_purchases))
sales.data$repeat_purchases_normalized <- scale(sales.data$repeat_purchases)
sales.data$State <- sapply(strsplit(as.character(sales.data$geography), ","), function(x) x[3])
```

Data Exploration

Sales Overview with respect to time(years)

I want to analyze customer purchasing behavior over time, specifically identifying peak periods of purchases and lows throughout the year. I utilized a line graph plot for easy visualization of the peaks and troughs in customer purchasing patterns.

```
purchase_frequency <- sales.data %>%
  group_by(order_date) %>%
  summarise(Orders = n())

ggplot(purchase_frequency, aes(x = order_date, y = Orders)) +
  geom_line() +
  labs(x = "Order Date", y = "Number of Orders", title = "Customer Purchase Frequency Over Time")
```

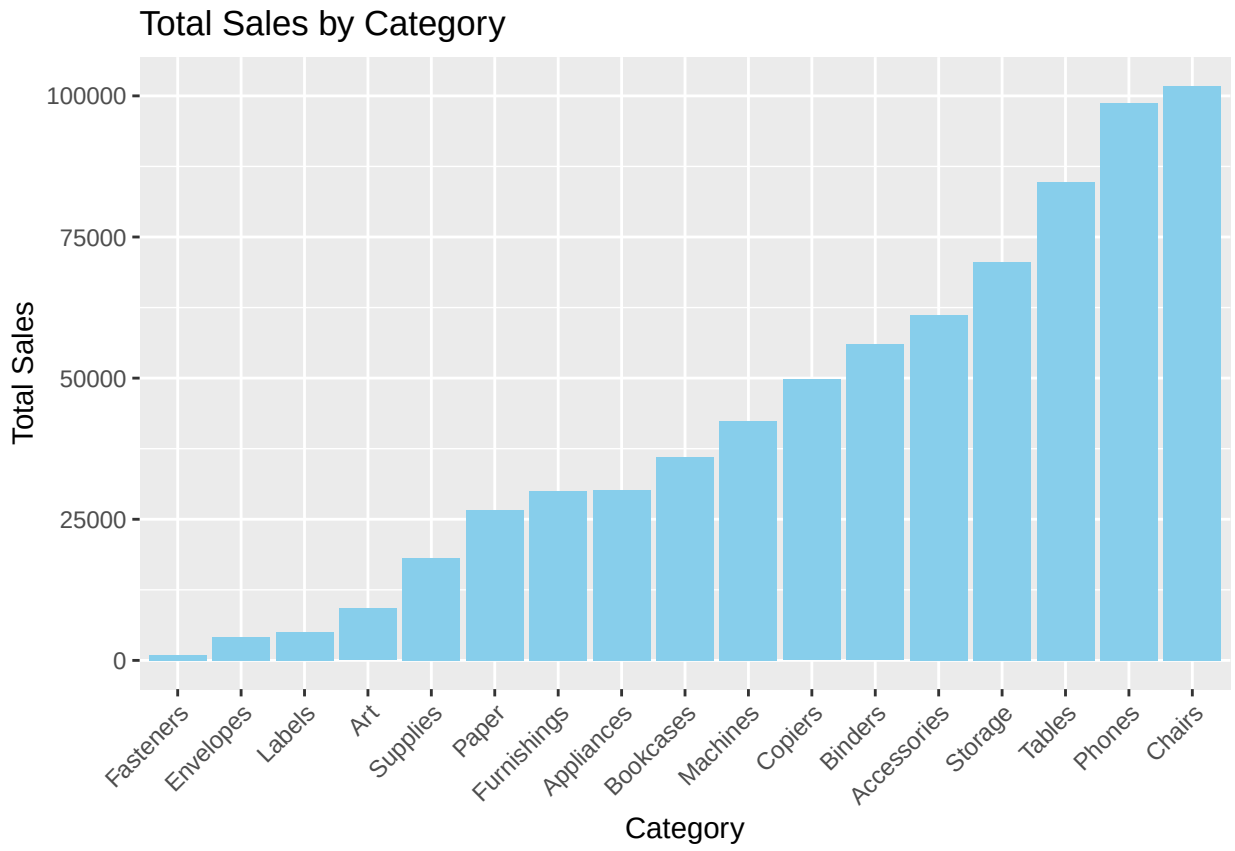


Data sorted with respect to category

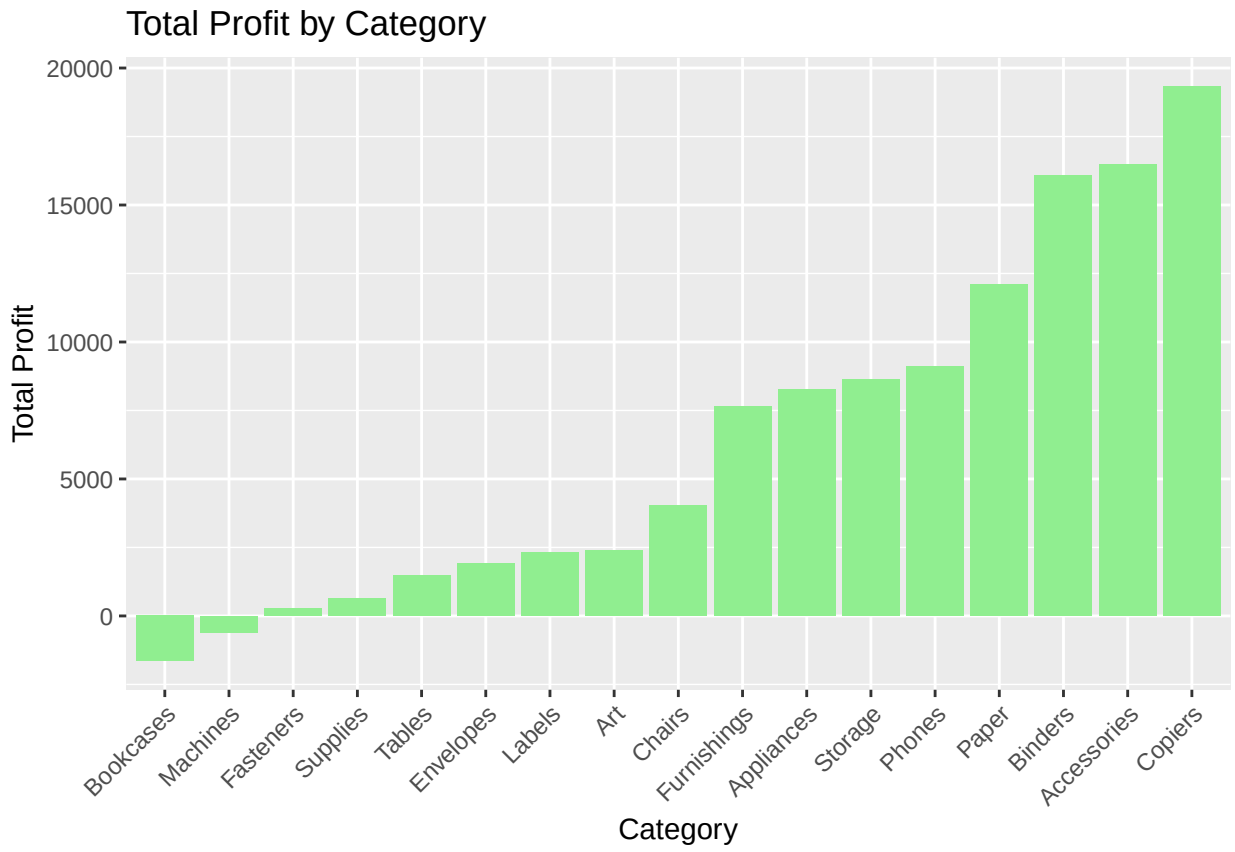
I sorted the data by category and proceeded to plot graphs depicting both sales and profit, providing insights into which categories are experiencing higher sales volume and which ones yield greater profit margins.

```
category_sales_profit <- sales.data %>%
  group_by(category) %>%
  summarise(Total_Sales = sum(sales),
            Total_Profit = sum(profit))


category_sales_profit <- category_sales_profit[order(category_sales_profit$Total_Sales, decreasing = TRU
```

```
ggplot(category_sales_profit, aes(x = reorder(category, Total_Sales), y = Total_Sales)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(x = "Category", y = "Total Sales", title = "Total Sales by Category") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

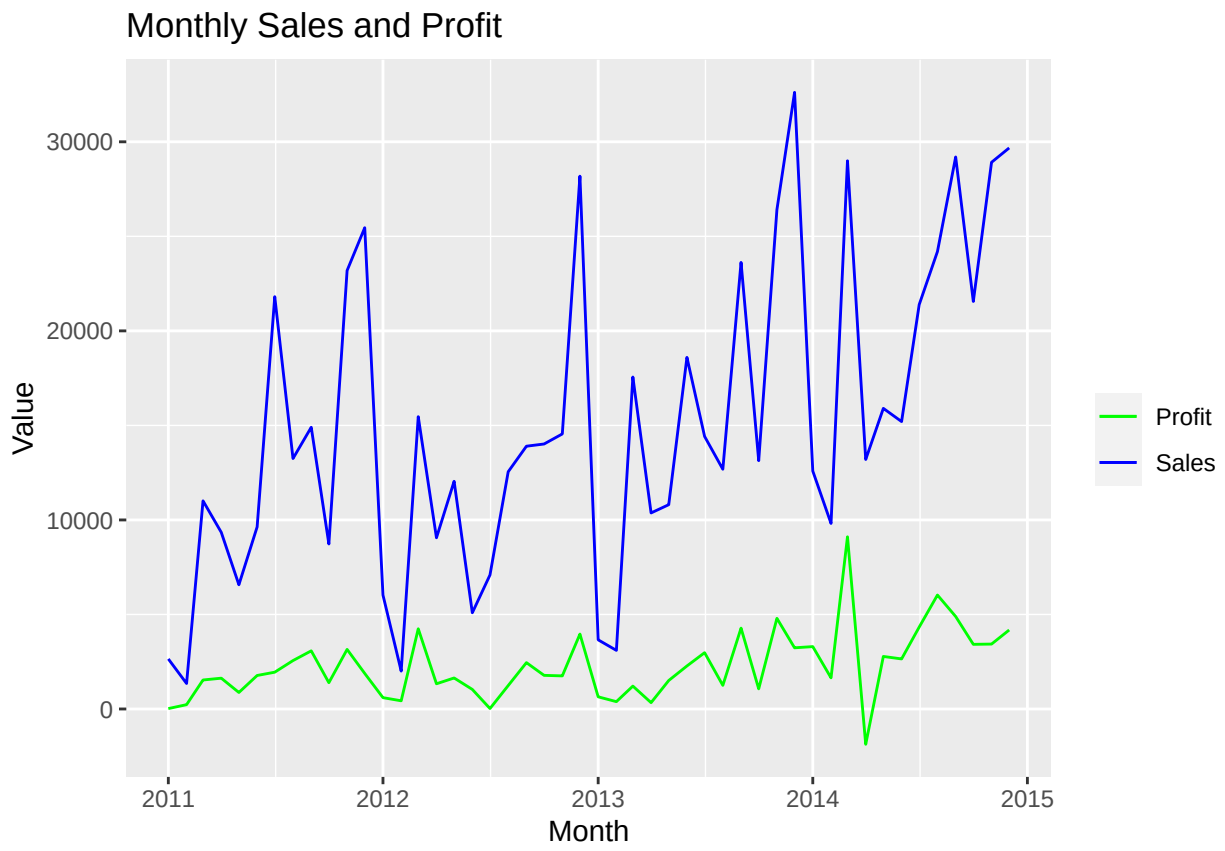



```
ggplot(category_sales_profit, aes(x = reorder(category, Total_Profit), y = Total_Profit)) +
  geom_bar(stat = "identity", fill = "lightgreen") +
  labs(x = "Category", y = "Total Profit", title = "Total Profit by Category") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Time Series Analysis

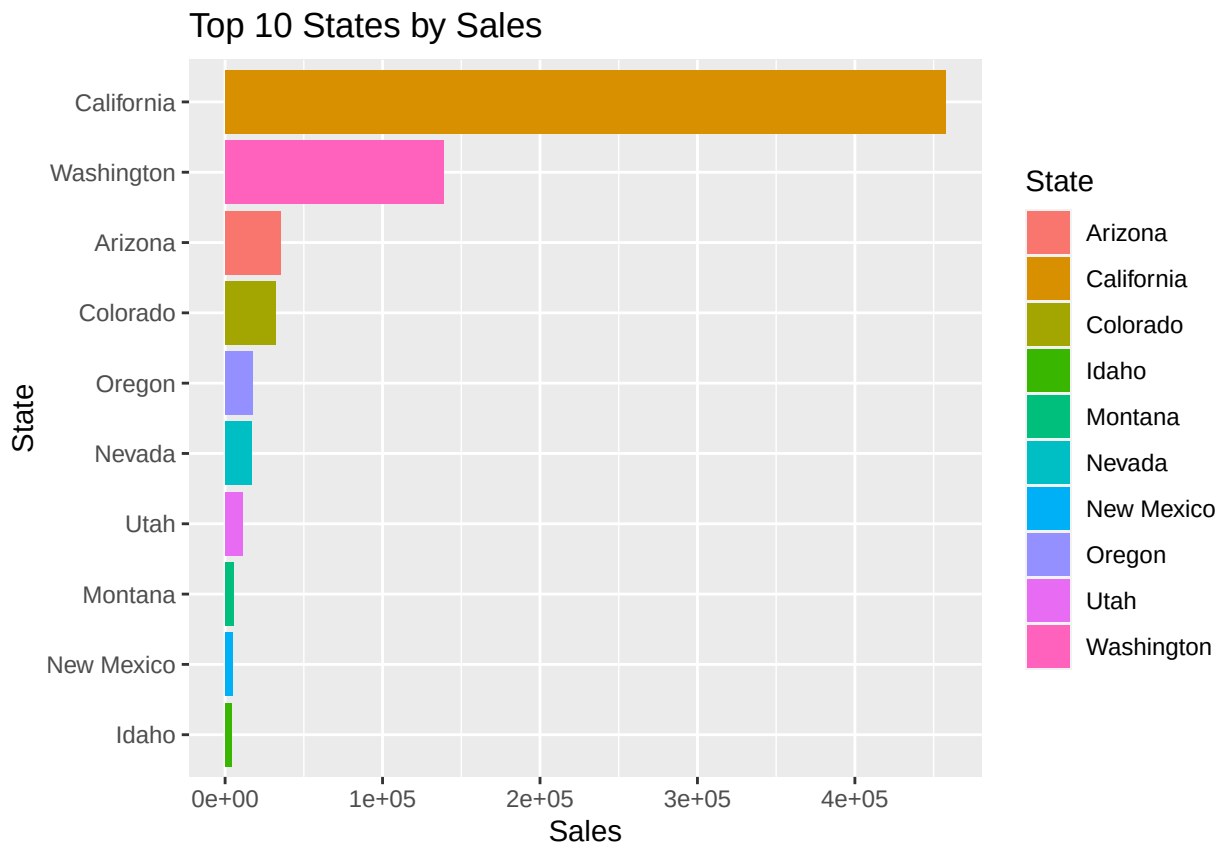
```
monthly_data <- sales.data %>%
  mutate(Month = floor_date(order_date, "month")) %>%
  group_by(Month) %>%
  summarise(sales = sum(sales), profit = sum(profit))

ggplot(monthly_data, aes(x = Month)) +
  geom_line(aes(y = sales, colour = "Sales")) +
  geom_line(aes(y = profit, colour = "Profit")) +
  labs(title = "Monthly Sales and Profit", x = "Month", y = "Value") +
  scale_colour_manual("", values = c("Sales" = "blue", "Profit" = "green"))
```



Sales have increased in the long run but have shown tumultuous behavior in the short term. On the other hand, profit has stayed somewhat steady with a spike in early 2014. Using this data we gain insight into how amazon has performed over time and we can use this data to make predictions about the future value of their company.

Geographical Analysis

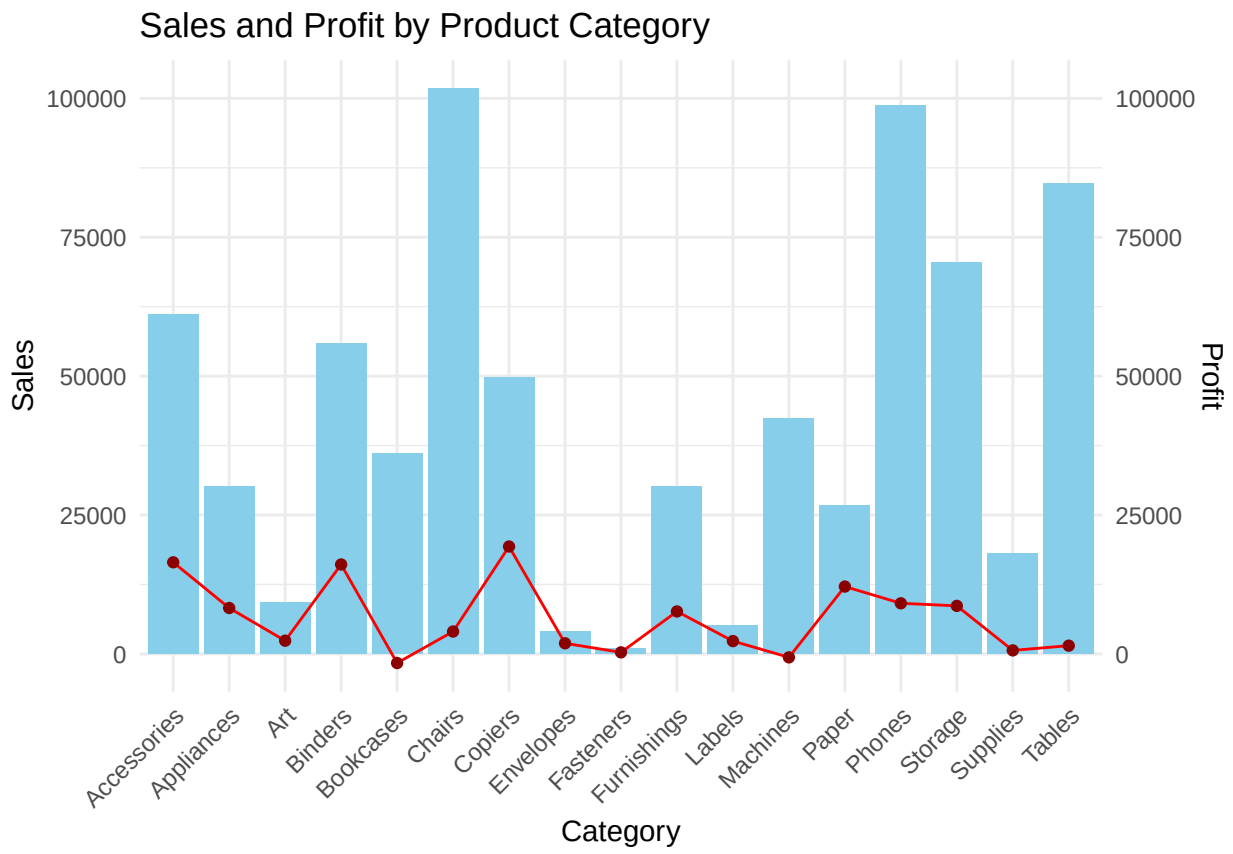
```
state_analysis <- sales.data %>%
  group_by(State) %>%
  summarise(sales = sum(sales), profit = sum(profit)) %>%
  arrange(desc(sales))

top_states <- head(state_analysis, 10)
ggplot(top_states, aes(x = reorder(State, sales), y = sales, fill = State)) +
  geom_bar(stat = "identity") +
  labs(title = "Top 10 States by Sales", x = "State", y = "Sales") +
  coord_flip()
```



Upon completing the geographical analysis, we can see California orders the most products from amazon of any other state by a wide margin. Washington comes in second with the rest of the states ordering roughly the same amount of products. This data could be used by amazon to create more facilities in California where the demand for products is higher.

Product Performance Analysis

```
category_performance <- sales.data %>%
  group_by(category) %>%
  summarise(sales = sum(sales), profit = sum(profit))

p <- ggplot(category_performance, aes(x=category)) +
  geom_bar(aes(y=sales), stat="identity", fill="skyblue") +
  theme_minimal() +
  labs(y="Sales", x="Category", title="Sales and Profit by Product Category") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

p + geom_line(aes(y=profit, group=1), color="red") +
  geom_point(aes(y=profit), color="darkred") +
  scale_y_continuous(sec.axis = sec_axis(~., name="Profit"))
```



The plot above illustrates the sales and profit by product category, showcasing a dual-axis view where the bars represent sales and the line with markers indicates profit for each category. This visualization enables us to see not just the raw sales numbers but also how those sales translate into profit across different product categories. From the plot, we can observe the performance of each category in terms of sales and profit, which might help in making informed decisions on which product categories to focus on or potentially expand.

Machine Learning Models

Linear Regression

```
# Feature Engineering
sales.data$order_day <- weekdays(as.Date(sales.data$order_date))
sales.data$repeat_purchases <- ave(sales.data$order_id, sales.data$email, FUN = length)

# Model Training
model <- lm(profit ~  category + repeat_purchases + quantity, data = sales.data)

# Model Evaluation
predicted_profit <- predict(model, newdata = sales.data)
mae <- mean(abs(predicted_profit - sales.data$profit))
mse <- mean((predicted_profit - sales.data$profit)^2)
rsquared <- summary(model)$r.squared

cat("Mean Absolute Error (MAE):", mae, "\n")
```

```
## Mean Absolute Error (MAE): 47.38676
```

```
cat("Mean Squared Error (MSE):", mse, "\n")
```

```
## Mean Squared Error (MSE): 25325.26
```

```
cat("R-squared:", rsquared, "\n")
```

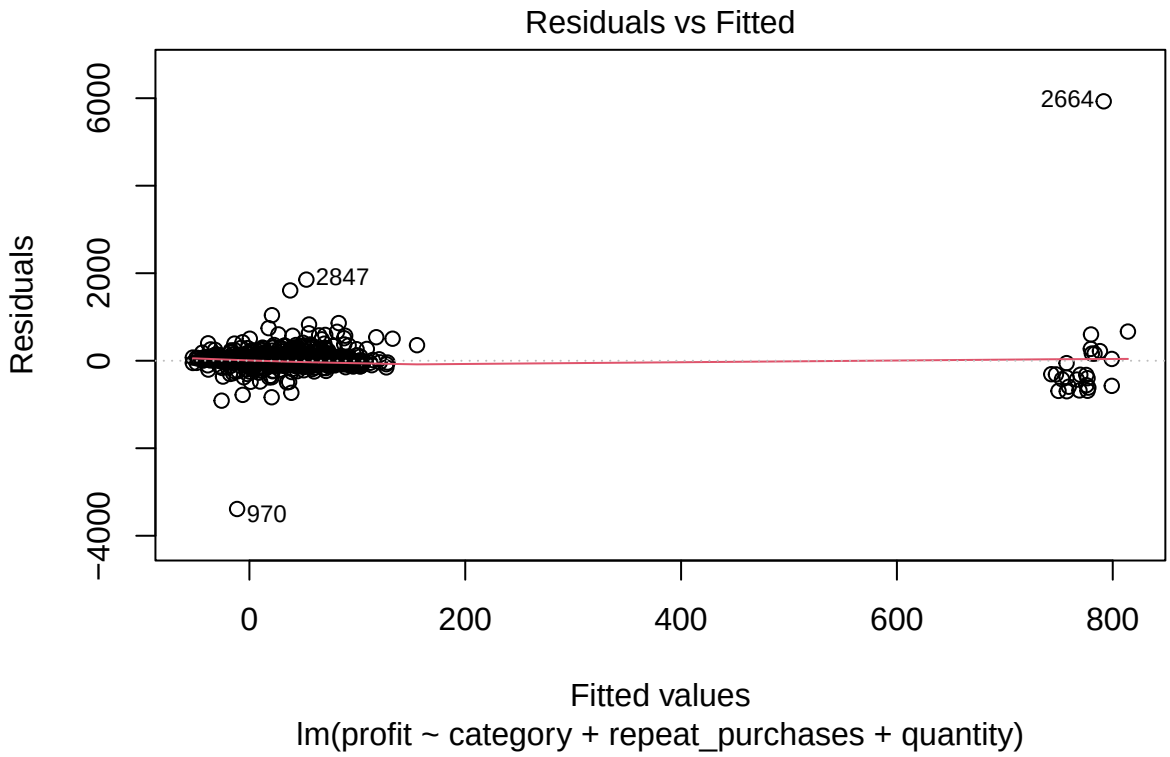
```
## R-squared: 0.164307
```

```
# Interpretation
kable(summary(model)$coefficients, caption = "Regression Coefficients")
```

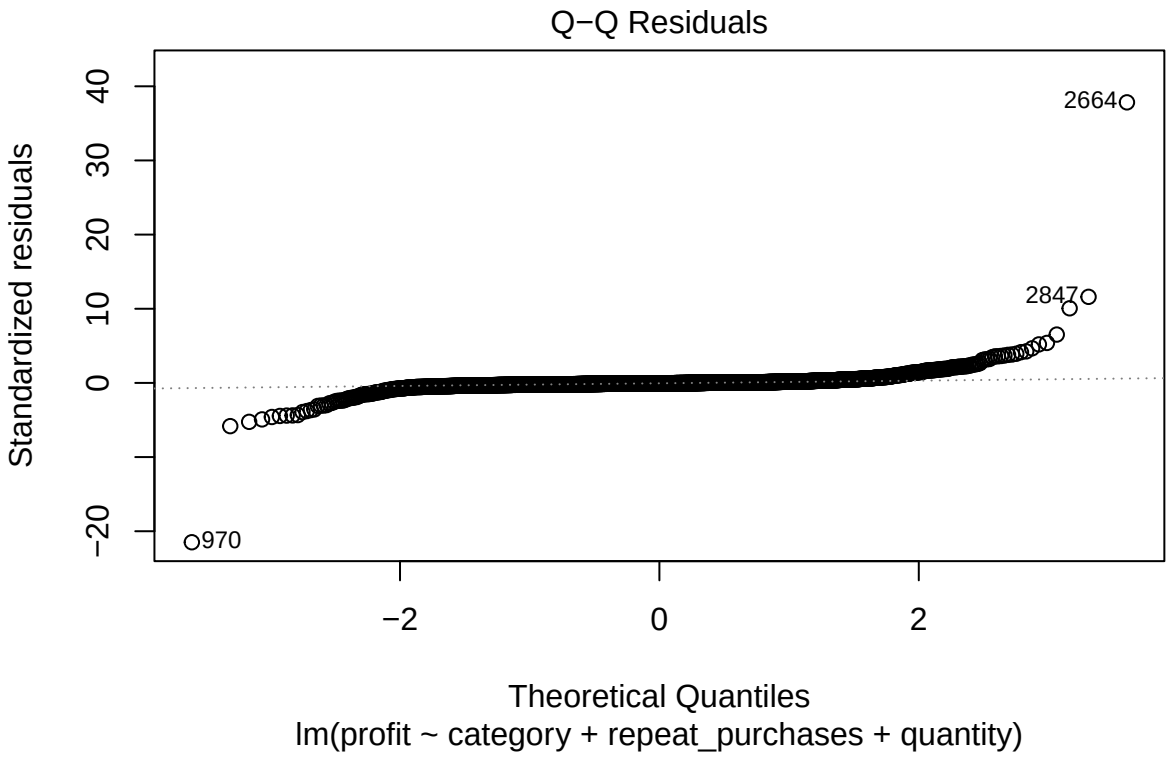
Table 3: Regression Coefficients

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	26.5882940	18.895389	1.4071314	0.1594866
categoryAppliances	-0.8435596	17.037757	-0.0495112	0.9605150
categoryArt	-53.9025739	14.268655	-3.7776913	0.0001612
categoryBinders	-29.2260596	12.431737	-2.3509232	0.0187879
categoryBookcases	-83.0452780	20.515185	-4.0479908	0.0000529
categoryChairs	-41.0297428	14.988042	-2.7374985	0.0062254
categoryCopiers	710.3784080	33.625121	21.1264194	0.0000000
categoryEnvelopes	-30.8478064	22.042189	-1.3994893	0.1617643
categoryFasteners	-57.5606642	21.396260	-2.6902208	0.0071780
categoryFurnishings	-37.7960763	13.598381	-2.7794542	0.0054772
categoryLabels	-44.9365532	17.923076	-2.5071897	0.0122192
categoryMachines	-80.8891720	27.573813	-2.9335505	0.0033752
categoryPaper	-35.8941122	12.529709	-2.8647204	0.0042013
categoryPhones	-30.9582143	13.883409	-2.2298712	0.0258259
categoryStorage	-31.5120225	14.025939	-2.2466962	0.0247281
categorySupplies	-53.2343036	21.762304	-2.4461704	0.0144923
categoryTables	-51.9427054	17.992553	-2.8869002	0.0039169
repeat_purchases10	-3.4583739	19.456929	-0.1777451	0.8589345
repeat_purchases11	-12.0763464	26.485723	-0.4559568	0.6484523
repeat_purchases12	1.8941407	23.256420	0.0814459	0.9350925
repeat_purchases13	-2.0803402	25.148779	-0.0827213	0.9340784
repeat_purchases14	-16.3831128	26.412705	-0.6202739	0.5351221
repeat_purchases15	7.6956155	21.239892	0.3623190	0.7171379
repeat_purchases16	0.1020580	32.226023	0.0031669	0.9974733
repeat_purchases17	-16.1055257	41.800177	-0.3852980	0.7000425
repeat_purchases19	24.3994737	39.854272	0.6122173	0.5404380
repeat_purchases2	-2.2160917	18.807304	-0.1178314	0.9062087
repeat_purchases23	3.4081178	36.790669	0.0926354	0.9261991
repeat_purchases24	11.1153357	36.116984	0.3077592	0.7582858
repeat_purchases3	8.7027888	18.009407	0.4832357	0.6289618
repeat_purchases4	6.2365259	17.796095	0.3504435	0.7260292
repeat_purchases5	5.4477132	17.873414	0.3047942	0.7605429
repeat_purchases6	24.6995606	17.479052	1.4130950	0.1577261
repeat_purchases7	17.9630356	17.534312	1.0244505	0.3057008
repeat_purchases8	14.7981302	17.940517	0.8248442	0.4095222
repeat_purchases9	1.0534967	18.077084	0.0582780	0.9535309
quantity	7.4896713	1.259961	5.9443683	0.0000000

```r
plot(model, which = 1)
```




```r
# Normal Q-Q Plot
plot(model, which = 2)
```



- The linear regression model uses `category`, `repeat_purchases`, and `quantity` as predictors to estimate `profit`.
- The Mean Absolute Error (MAE) is 47.39, indicating that the model's predictions are, on average, $47.39 away from the actual profit values.
- The Mean Squared Error (MSE) is reported at 25325.26, suggesting that the model's predictions are quite variable and may be influenced by outliers or extreme values.
- The R-squared value of the model is 0.1643, which means that about 16.43% of the variation in `profit` is explained by the model. This is a relatively low value, implying that the model might not be capturing all the factors that influence profit.
- Diagnostic plots, such as Residuals vs Fitted and Q-Q plots, suggest the presence of non-linearity or outliers within the data that could be affecting model accuracy.

Neural Network with k fold

```
data_for_nn <- model.matrix(~ category + order_day + repeat_purchases_normalized - 1, data = sales.data
target <- sales.data$profit
set.seed(123)
k <- 5
folds <- createFolds(target, k = k, list = TRUE, returnTrain = FALSE)

# Initialize variables to store performance metrics
maes <- numeric(k)
mses <- numeric(k)

# Loop through each fold
for(i in 1:k) {
```

```r
  # Split the data into training and testing based on folds
  test_indices <- folds[[i]]
  train_indices <- setdiff(1:nrow(data_for_nn), test_indices)

  train_data <- data_for_nn[train_indices, ]
  test_data <- data_for_nn[test_indices, ]

  train_target <- target[train_indices]
  test_target <- target[test_indices]

  # Train the model
  nn_model <- nnet(train_data, train_target, size = 5, decay = 0.000005, linout = TRUE, maxit = 3000)
  # Make predictions
  predictions <- predict(nn_model, test_data)

  # Calculate and store the performance metrics for this fold
  maes[i] <- mean(abs(predictions - test_target))
  mses[i] <- mean((predictions - test_target)^2)
}
```

```
## # weights:  131
## initial  value 44034439.544389
## iter  10 value 38668042.136634
## iter  20 value 37581046.933876
## iter  30 value 36980310.173474
## iter  40 value 36451514.633498
## iter  50 value 36343386.665609
## iter  60 value 36254029.462331
## iter  70 value 36206925.418374
## iter  80 value 36069820.227878
## iter  90 value 36032636.210982
## iter 100 value 35961629.878108
## iter 110 value 35837620.249765
## iter 120 value 35828173.486223
## iter 130 value 35820243.568291
## iter 140 value 35807989.925279
## iter 150 value 35774863.628332
## iter 160 value 35770467.613853
## iter 170 value 35768685.666043
## iter 180 value 35759905.713383
## iter 190 value 35756195.924754
## iter 200 value 35751596.491510
## iter 210 value 35746476.521137
## iter 220 value 35729437.875553
## iter 230 value 35729071.968858
## iter 240 value 35729041.503054
## final  value 35729033.627083
## converged
## # weights:  131
## initial  value 88105695.109307
## iter  10 value 73081265.921701
## iter  20 value 71332919.637096
## iter  30 value 70562220.212615
## iter  40 value 64298637.753173
```

```
## iter  50 value 61183379.826802
## iter  60 value 59801250.983643
## iter  70 value 59244646.718026
## iter  80 value 58442830.344558
## final  value 58442771.485213
## converged
## # weights:  131
## initial  value 93857949.853849
## iter  10 value 84554336.669497
## iter  20 value 77336217.379974
## iter  30 value 74796284.330562
## iter  40 value 74091756.570071
## iter  50 value 73775851.316917
## iter  60 value 73507238.967081
## iter  70 value 73253994.952431
## iter  80 value 73172149.844058
## iter  90 value 73041587.423796
## iter 100 value 72986076.749067
## iter 110 value 72731365.731512
## iter 120 value 72271071.783418
## iter 130 value 71762482.877432
## iter 140 value 71564230.679209
## iter 150 value 71472884.340652
## iter 160 value 71280455.611593
## iter 170 value 71204738.325665
## iter 180 value 71171291.961081
## iter 190 value 71153770.245486
## iter 200 value 71151108.985821
## iter 210 value 71122978.054834
## iter 220 value 71109831.402013
## iter 230 value 71105524.680354
## iter 240 value 71103682.350413
## iter 250 value 71094988.945946
## iter 260 value 71064970.588352
## iter 270 value 71063336.941611
## iter 280 value 71061782.114738
## iter 290 value 71051786.159252
## iter 300 value 71042157.943859
## iter 310 value 71038737.830964
## iter 320 value 71037012.079902
## final  value 71036755.921848
## converged
## # weights:  131
## initial  value 94524748.270760
## iter  10 value 81931623.186320
## iter  20 value 79045477.417500
## iter  30 value 75557504.591133
## iter  40 value 74809499.076465
## iter  50 value 74590386.452420
## iter  60 value 74355771.048946
## iter  70 value 74248505.107773
## iter  80 value 74174005.656226
## iter  90 value 74117290.757466
## iter 100 value 74115054.655131
```

```
## final  value 74115030.543421
## converged
## # weights:  131
## initial  value 82663911.401256
## iter  10 value 69867185.760264
## iter  20 value 58737178.045564
## iter  30 value 57107648.200548
## iter  40 value 56831967.208684
## iter  50 value 56791630.558803
## iter  60 value 56775084.246067
## final  value 56772514.605100
## converged
```

```r
# Calculate the average performance across all folds
avg_mae <- mean(maes)
avg_mse <- mean(mses)

cat("Average Mean Absolute Error (MAE) across all folds:", avg_mae, "\n")
```

```
## Average Mean Absolute Error (MAE) across all folds: 50.21544
```

```r
cat("Average Mean Squared Error (MSE) across all folds:", avg_mse, "\n")
```

```
## Average Mean Squared Error (MSE) across all folds: 31663.92
```

- The neural network model uses `category`, `order_day`, and `repeat_purchases_normalized` from `sales.data` as inputs.
- The model's performance was validated using 5-fold cross-validation to assess consistency across different data subsets.
- **Average Mean Absolute Error (MAE):** The MAE across all folds is approximately $50.22, indicating the average deviation of the neural network's predictions from the actual profit values.
- **Average Mean Squared Error (MSE):** The MSE is 31663.92, suggesting there is significant variance in the model's predictions, with higher errors for some predictions.
- The training process converged consistently across different folds and initial values, demonstrating the stability of the model.
- The model's reasonably accurate predictions, given the MAE and MSE, suggest a potential for improvement.

Regression using Elastic Net model

```r
# Data
data_matrix <- model.matrix(~ year + month + day_of_week + category + sales_normalized + quantity_normal
profit_vector <- sales.data$profit

# Elastic Net Regression
set.seed(123)
cv_elasticnet <- cv.glmnet(data_matrix, profit_vector, alpha = 0.5)
elasticnet_model <- glmnet(data_matrix, profit_vector, alpha = 0.5, lambda = cv_elasticnet$lambda.min)

# Predictions
predictions_elasticnet <- predict(elasticnet_model, s = cv_elasticnet$lambda.min, newx = data_matrix)

# Calculate MAE and MSE
mae_elasticnet <- mean(abs(predictions_elasticnet - profit_vector))
mse_elasticnet <- mean((predictions_elasticnet - profit_vector)^2)
cat("Elastic Net - Mean Absolute Error (MAE):", mae_elasticnet, "\n")
```

```
## Elastic Net - Mean Absolute Error (MAE): 39.3798
```

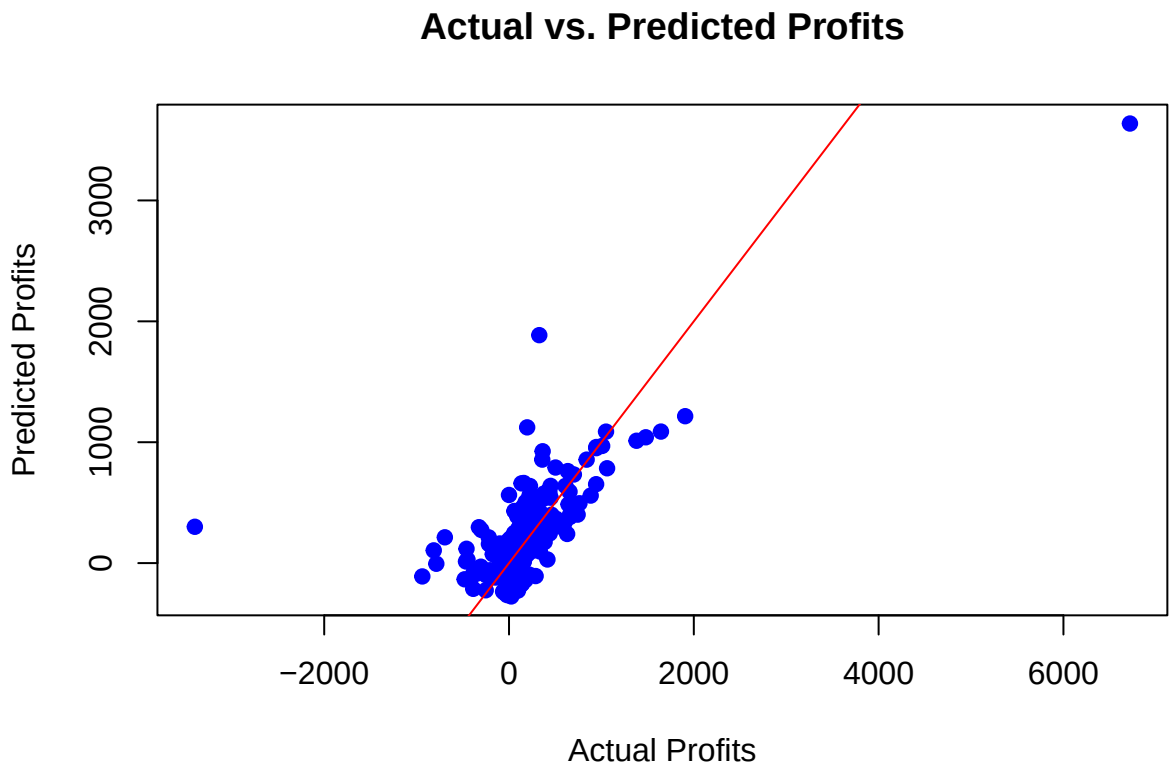
```
cat("Elastic Net - Mean Squared Error (MSE):", mse_elasticnet, "\n")
```

```
## Elastic Net - Mean Squared Error (MSE): 13956.61
```

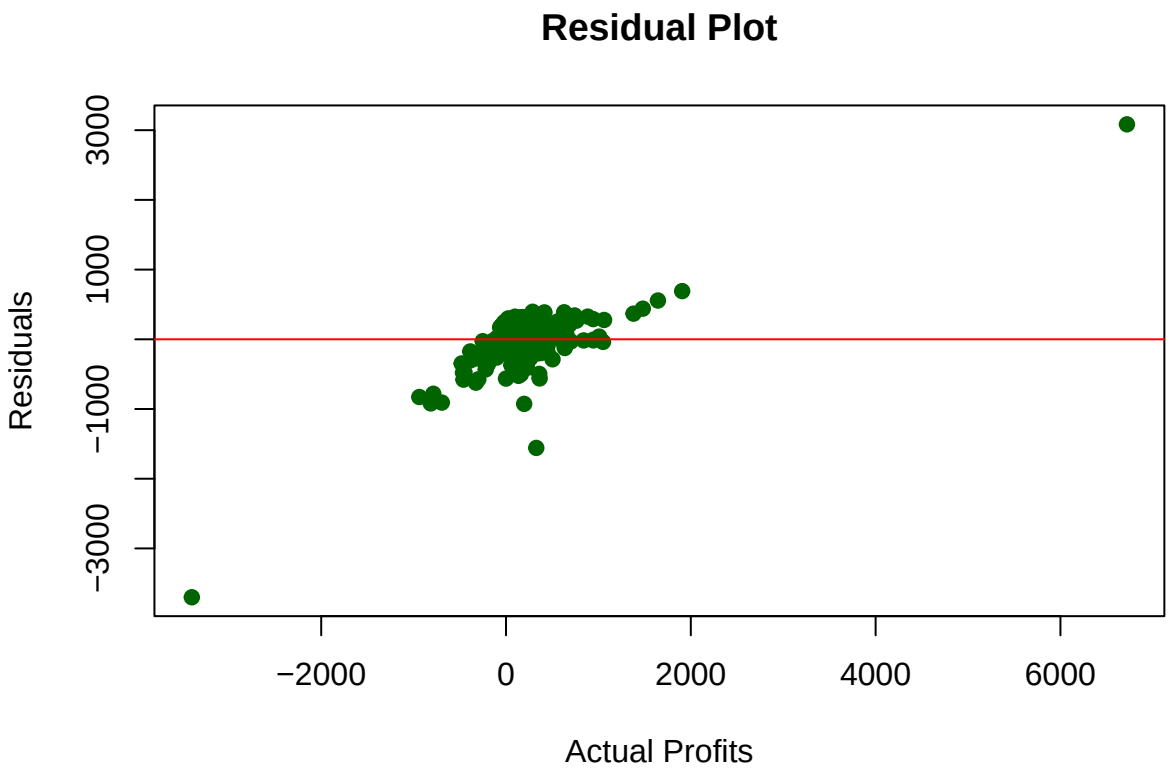
```
actuals <- profit_vector

# Calculate R-squared
ss_total <- sum((actuals - mean(actuals))^2)
ss_res <- sum((actuals - predictions_elasticnet)^2)
r_squared <- 1 - (ss_res / ss_total)
cat("R-squared:", r_squared, "\n")
```

```
## R-squared: 0.5394542
```

```
# Actual VS Predicted Plot
plot(actuals, predictions_elasticnet, main = "Actual vs. Predicted Profits",
     xlab = "Actual Profits", ylab = "Predicted Profits", pch = 19, col = "blue")
abline(0, 1, col = "red") # Diagonal line indicating perfect predictions
```




```
# Residual Plot
residuals <- actuals - predictions_elasticnet
plot(actuals, residuals, main = "Residual Plot",
     xlab = "Actual Profits", ylab = "Residuals", pch = 19, col = "darkgreen")
abline(h = 0, col = "red") # Horizontal line at 0 indicating no error
```



- The Elastic Net regression model, with an alpha of 0.5, demonstrates a balance between Lasso and Ridge regression characteristics. It was trained on an extensive set of predictors derived from `sales.data`.
- The model achieved a Mean Absolute Error (MAE) of 39.3798, indicating that the predictions are on average approximately $39.38 away from the actual profit values.
- A Mean Squared Error (MSE) of 13956.61 was observed, which represents the average of the squares of the errors. The lower MSE, as compared to the neural network model, suggests improved prediction accuracy.
- The model accounts for approximately 53.945% of the variability in the profit data, as indicated by an R-squared value of 0.5394542. This represents a moderate to strong fit and a substantial improvement over the linear regression model.
- The "Actual vs. Predicted Profits" plot shows that while many predictions are close to the actual values (as indicated by the proximity to the red diagonal line), there is still some variance, especially with higher profit values. The "Residual Plot" further suggests that residuals are not randomly distributed around zero, implying the presence of outliers.

Random Forest

```
set.seed(123)
rf_model <- randomForest(x = data_matrix, y = profit_vector, ntree = 500)

#Predictions
predictions_rf <- predict(rf_model, newdata = data_matrix)

#Calculate MAE and MSE
mae_rf <- mean(abs(predictions_rf - profit_vector))
mse_rf <- mean((predictions_rf - profit_vector)^2)
```

```r
cat("Random Forest - Mean Absolute Error (MAE):", mae_rf, "\n")
```

```
## Random Forest - Mean Absolute Error (MAE): 13.73329
```

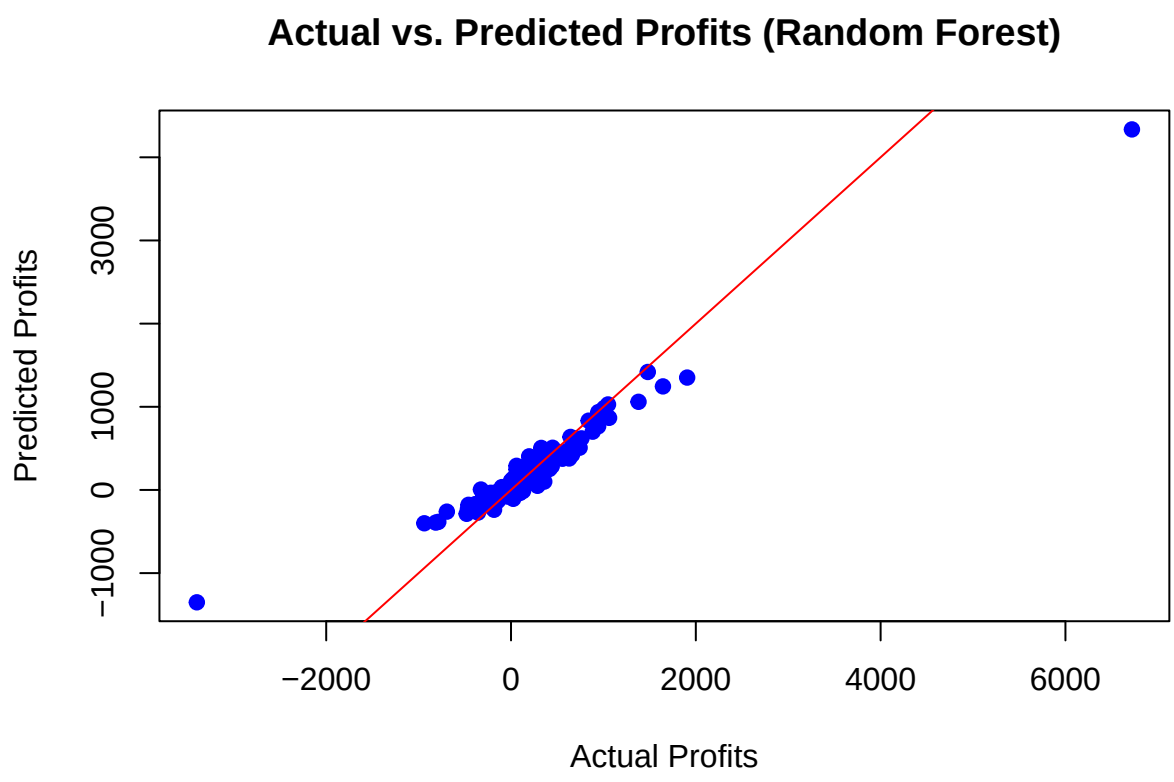
```r
cat("Random Forest - Mean Squared Error (MSE):", mse_rf, "\n")
```

```
## Random Forest - Mean Squared Error (MSE): 4395.276
```

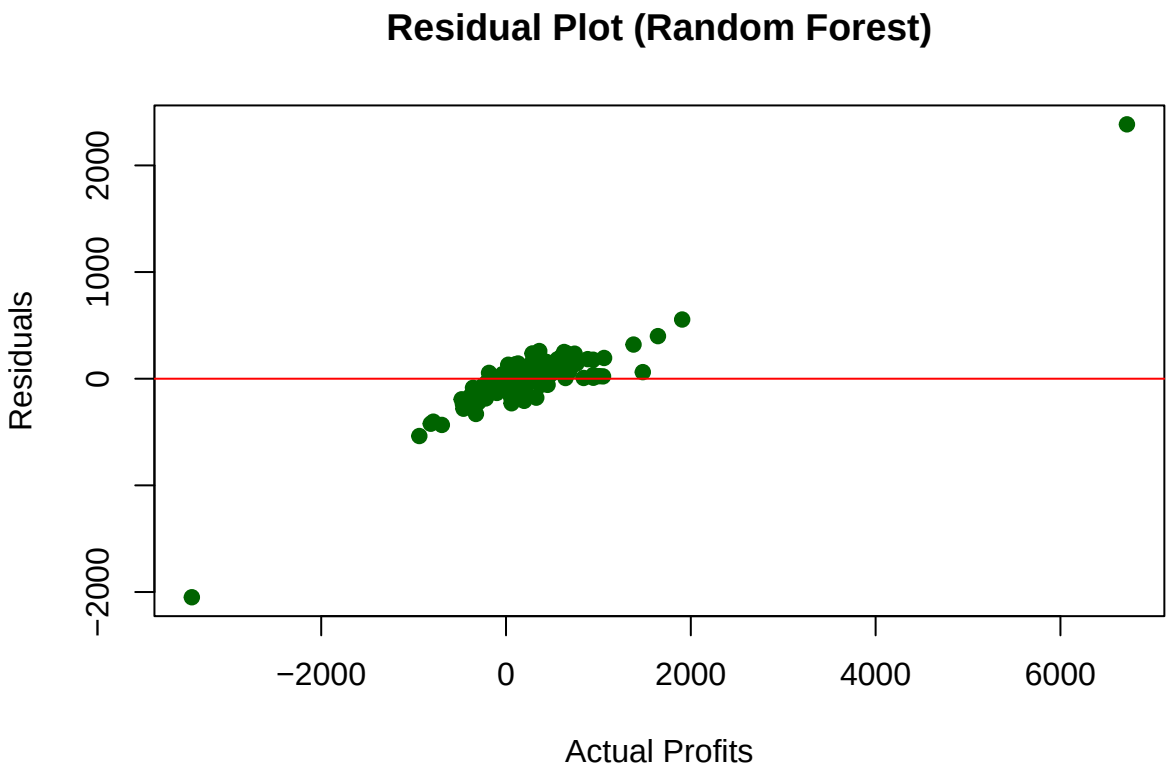
```r
actuals <- profit_vector

#Calculate R-squared
ss_total <- sum((actuals - mean(actuals))^2)
ss_res <- sum((actuals - predictions_rf)^2)
r_squared_rf <- 1 - (ss_res / ss_total)
cat("R-squared:", r_squared_rf, "\n")
```

```
## R-squared: 0.854963
```

```r
#Actual VS Predicted Plot
plot(actuals, predictions_rf, main = "Actual vs. Predicted Profits (Random Forest)",
     xlab = "Actual Profits", ylab = "Predicted Profits", pch = 19, col = "blue")
abline(0, 1, col = "red") # Diagonal line indicating perfect predictions
```




```r
#Residual Plot
residuals_rf <- actuals - predictions_rf
plot(actuals, residuals_rf, main = "Residual Plot (Random Forest)",
     xlab = "Actual Profits", ylab = "Residuals", pch = 19, col = "darkgreen")
abline(h = 0, col = "red")
```



- The Random Forest model, utilizing 500 trees, shows strong predictive performance with an R-squared value of 0.854963. This indicates that the model explains approximately 85.5% of the variance in profit data, which is quite high and suggests a good fit.
- A Mean Absolute Error (MAE) of 13.73329 suggests that the model's predictions are, on average, about $13.73 away from the actual profit values, demonstrating a high level of accuracy.
- The Mean Squared Error (MSE) of 4395.276 indicates that the model has a low average error squared, which confirms the accuracy seen in the MAE.
- The "Actual vs. Predicted Profits (Random Forest)" plot illustrates a tight clustering of points along the red line that indicates perfect predictions, demonstrating the model's accuracy in predicting profits.
- In the "Residual Plot (Random Forest)", most residuals are distributed close to the horizontal line at 0, with fewer large residuals compared to previous models, suggesting that the Random Forest model has a consistent prediction quality across the range of actual profits.